Introduction

This assignment asks you to design, implement and explore simple class hierarchies, abstract classes and interfaces.

Create two projects, one for part (a) and another for part (b).

(a) Shape up

Note: this assignment has nothing to do with drawing graphics in Java... everything is done on the console as usual.

(1) Design a class hierarchy to include classes *Shape*, *Rectangle* (with *int* sides, *width* and *height*), *Circle* (with *int radius*) and *Square* (with *int side*). Shape should be abstract with method *double getArea()* and Square should inherit *getArea()* from Rectangle.

(2) Create another class, *ShapeContainer*, to hold a set of shapes. It should have methods *void add(Shape s)* and *double getArea()* and *String toString()*. Write a *ShapeTester* class with a menu that allows the user to create an empty set of shapes (ShapeContainer), add as many circle and rectangle shapes to it as they wish, compute & print out the total surface area of the entire set of shapes, and print out information about all of the shapes in the container by calling the toString() method for each shape. Experiment. Try to predict what would happen when you (i) comment out the getArea() method of the Circle class, and (ii) also make the Circle class abstract, before finally (iii) creating an instance of the (now abstract) Circle class to add to the shapes collection. Test your predictions.

(3) The customer is impressed with your work so far, and so asks you to extend the program. They want shapes to be locatable (i.e. to have an *x, y* location, and *getX()*, *getY()* and *setLocation(x, y)* methods). As a good designer you decide to first create a *Locatable* interface with these methods, then have the *Shape* class implement it. In this way all shapes automatically become locatable.

(4) Impressed, the customer wants even more! This time they ask for shapes to be *Selectable*, so you again start by creating a Java interface, having *boolean getSelected()* and *setSelected(boolean)* and *Shape contains(int x, int y)*. Unfortunately, this time you are *not allowed* to change the *Shape* class. Modify your other classes so that each shape added to the *ShapeContainer* is *Selectable*. Change the *toString()* methods of each shape class so they show whether the shape is selected or not. Add another option to your ShapeTester menu that allows the user to find the first Shape that contains a given x, y point and, afterwards, toggle its selected state. Provide another menu option that removes all selected shapes from the set of shapes. Good design practice suggests you should ask the *ShapeContainer* object to do the work of finding the first Shape containing the given point and of removing all selected shapes (rather than trying to do the work yourself in the ShapeTester class, which might/would require knowledge of the insides of the *ShapeContainer* class).

(b) ...coming soon.
